You have **1** free member-only story left this month. Sign up for Medium and get an extra one

Simple Mortgage Calculator With Python and Excel

Automate your real estate analysis with Pandas and Numpy

Pendora Follow
May 23, 2020 · 7 min read ★

Real estate analysis isn't complete without automating the calculation of mortgage payments. We'll look to create our mortgage calculator from scratch in both Excel and Python. By the end of this article, you'll know exactly how to conduct your analysis for different types of loans and mortgages when you're confronted by advisers who are compensated based on your business with them. Choose the best deal for you and your family, not the best deal for the bank providing the mortgage.

For business owners looking for mortgages to take on for business purposes, this may be helpful in your search for the best property. Feel free to take this calculator and add it to your real estate analysis toolkit. If you follow along for both, you'll have a deep understanding of how things work, but feel free to jump to the Excel or Python sections however you please.

Photo by Vita Vilcina on Unsplash

Excel Mortgage Amortization Schedule

Starting with our assumptions, we need to set up cells with every variable in the mortgage calculation. The interest rate, amortization period, payments per year, and the mortgage amount (principal). We'll use a mortgage to be paid off in 30 years, and monthly payments (12 per year). Sometimes mortgages will need to be paid bi-weekly (24 per year), and the compounding would look a little differently. Usually, lenders will provide the interest rate as an annual rate, so all we need to do is divide by the number of payments per year.

Basic Assumptions	
Annual interest rate	4%
Year	30
Payments per year	12
Monthly interest rate	0.333%
Total mortgage amount	$400,000
Property value	$750,000

Next, let's set up our Excel table, including all of the aspects of the calculation.

Mortgage Amortization Schedule (Monthly)					
Period	Beginning Balance	Total Payment	Principal Paid	Interest Paid	Ending Balance
0	$0	$0	$0	$0	$250,000
1					
2					
3					
4					
5					

Table 0: Beginning Amortization Table Template

Keep in mind that each period represents one month, and although we'll only be showing the first 5 periods, we need to copy the formulas down to 360 months (30 years). The first row (period 0) has no payments because we just took out the mortgage, the first payment occurs in period 1.

The beginning balance of period 1 should be linked to the ending balance of the period before.

Mortgage Amortization Schedule (Monthly)					
Period	Beginning Balance	Total Payment	Principal Paid	Interest Paid	Ending Balance
0	$0	$0	$0	$0	$250,000
1	=G13				
2	$0				
3	$0				
4	$0				
5	$0				

Table 1: Linking Beginning Balance to Previous Ending Balance

Next, we can use a powerful Excel function **PMT** to simplify the calculation of monthly payments. A bit more finance knowledge is required, but the mathematical formula to do so would be:

$$M = \frac{P\left(\frac{r}{12}\right)\left(1+\frac{r}{12}\right)^{n}}{\left(1+\frac{r}{12}\right)^{n}-1}$$

M = monthly payment

P = amount borrowed

r = annual interest rate

n = number of monthly payments

Feel free to dig into this formula more if you like, but the PMT function reduces all of the calculating for us.

Payment = PMT(Monthly Interest Rate, Periods, Mortgage)

In our case, monthly interest is 4% divided by 12 payments per year, 360 periods, and $400,000 mortgage. Since these things don't change, we need to use absolute cell references to ensure the proper cells are referenced.

Basic Assumptions	
Annual interest rate	4%
Year	30
Payments per year	12
Monthly interest rate	0.333%
Total mortgage amount	$400,000
Property value	$700,000

Mortgage Amortization Schedule (Monthly)						
Period	Beginning Balance	Total Payment	Principal Paid	Interest Paid	Ending Balance	Cumulative Interest
0	$0	$0	$0	$0	$400,000	$0
1	$400,000	=-PMT(E6,E4*E5,C14)				

I've also calculated the interest paid using the monthly interest rate multiplied by the

beginning balance. To see exactly how cumulative interest we're paying overtime, I've also added a column for cumulative interest, which just takes the interest paid for the current period and cumulative balance the period before.

Remember that each monthly payment includes a split between the principal paid off, as well as the interest. Lock the interest rate in place so things don't change when we

Period	Beginning Balance	Total Payment	Principal Paid	Interest Paid	Ending Balance	Cumulative Interest
0	0	0	0	0	=E7	0
=B13+1	=G13	=-PMT(E6,E4*E5,C14)	=D14-F14	=C14*E6	=C14-E14	=F14+H13

balance should just be the beginning balance minus the principal paid:

Now we just need to copy the row downward to complete the table.

Mortgage Amortization Schedule (Monthly)

Period	Beginning Balance	Total Payment	Principal Paid	Interest Paid	Ending Balance	Cumulative Interest
0	$0	$0	$0	$0	$400,000	$0
1	$400,000	$1,910	$576	$1,333	$399,424	$1,333
2	$399,424	$1,910	$578	$1,331	$398,845	$2,665
3	$398,845	$1,910	$580	$1,329	$398,265	$3,994
4	$398,265	$1,910	$582	$1,328	$397,683	$5,322
5	$397,683	$1,910	$584	$1,326	$397,099	$6,647
355	$11,325	$1,910	$1,872	$38	$9,454	$287,383
356	$9,454	$1,910	$1,878	$32	$7,575	$287,415
357	$7,575	$1,910	$1,884	$25	$5,691	$287,440
358	$5,691	$1,910	$1,891	$19	$3,800	$287,459
359	$3,800	$1,910	$1,897	$13	$1,903	$287,472
360	$1,903	$1,910	$1,903	$6	($0)	$287,478

Completed Amortization Table with First and Last 5 Rows Shown

Perfect, now we've completed our very own mortgage amortization calculator. If we add the sum of every amount in the *principal paid* column, we should arrive at $400,000. Looking at our *cumulative interest* column, we see that the *total cumulative interest paid* on the $400,000 mortgage is $287,478 over 30 years. Notice that *interest paid* every period declines, since the balance interest is charged on is slowly repaid over time. Near the end of the mortgage, nearly all of the payment goes to pay off the actual mortgage.

Python Mortgage Calculator

To begin, let's first see how we can use a simple Python script to check our answer.

Notice that this code is stinky, and is pretty difficult to follow. We'll be using a much better library and function later.

```
[48]: # Mortgage Calculator

def mortgagecalculator (mortgage, years, interest):
    interest = interest / 100
    nper = years*12
    interest_monthly = interest/12
    numerator = interest_monthly*((1+interest_monthly)**nper)
    denominator = (1+interest_monthly)**nper-1
    payment = float("{0:.2f}".format(mortgage*numerator/denominator))
    return(payment)

[49]: mortgagecalculator(400000,30,4)

[49]: 1909.66
```

Expected Code Output

This code puts the mortgage payment calculus into Python, and we can see that our output matches our own Excel. **Numpy** helps us with this calculation in a similar way that Excel did, so let's get fancy.

Importing Libraries

As you can probably tell based on our Excel calculator, to recreate the same thing in Python means we should probably use **Pandas.**

```
import pandas as pd
import numpy as np
from datetime import date
```

Numpy helps us with the calculations, and Pandas will help us with creating the row-by-row calculator. Let's first start by setting up the same assumptions. Assume that we take on the mortgage on January 1, 2021.

```
interest = 0.04
years = 30
payments_year = 12
mortgage = 400000
```

```
start_date = (date(2021, 1, 1))
```

Here's the Numpy equivalent of the PMT formula in Excel.

```
[59]: pmt = -1 * np.pmt(interest/12, years*payments_year, mortgage)

[60]: pmt

[60]: 1909.6611818618153
```

Since we have two anchor points to check this answer with, it's good that it matches. Let's also calculate the first interest payment, as well as the first principal payment. We didn't use these functions in Excel earlier (IPMT and PPMT), because we wanted to get a foundation of how things work, but now that we know how they work, we can use shortcuts instead.

```
[63]: # Monthly Payment
pmt = -1 * np.pmt(interest/12, years*payments_year, mortgage)

# Interest Payment
ipmt = -1 * np.ipmt(interest/payments_year, 1, years * payments_year, mortgage)

# Principal Payment
ppmt = -1 * np.ppmt(interest/payments_year, 1, years * payments_year, mortgage)

[64]: print(pmt, ipmt, ppmt)

1909.6611818618153 1333.3333333333335 576.3278485284818
```

Double-check with our Excel calculator's first row and we'll see that things match perfectly.

Awesome. Let's now start to build our DataFrame which will house all of this data.

```
rng = pd.date_range(start_date, periods=years * payments_year,
freq='MS')
rng.name = "Payment Date"

df = pd.DataFrame(index=rng, columns=['Payment', 'Principal Paid',
'Interest Paid', 'Ending Balance'], dtype='float')
```

```
df.reset_index(inplace=True)
df.index += 1
df.index.name = "Period"
```

We take our start date and increment each by one month. We also have an index called “Period” which matches the month of payment of our Excel calculator. Note that there is no 0th period in this code.

```
[41]: df
```

[41]:

Period	Payment Date	Payment	Principal Paid	Interest Paid	Ending Balance
1	2020-01-01	NaN	NaN	NaN	NaN
2	2020-02-01	NaN	NaN	NaN	NaN
3	2020-03-01	NaN	NaN	NaN	NaN
4	2020-04-01	NaN	NaN	NaN	NaN
5	2020-05-01	NaN	NaN	NaN	NaN
...	...	...	...	...	...
356	2049-08-01	NaN	NaN	NaN	NaN
357	2049-09-01	NaN	NaN	NaN	NaN
358	2049-10-01	NaN	NaN	NaN	NaN
359	2049-11-01	NaN	NaN	NaN	NaN
360	2049-12-01	NaN	NaN	NaN	NaN

360 rows × 5 columns

Not bad, and we can easily populate a few of the columns right away with the power of Numpy. The only hard column will be *Ending Balance* because I haven’t found a formula to calculate this for us yet and we’ll need to use subtraction similar to Excel.

```
[78]: df["Payment"] = -1 * np.pmt(interest/12, years*payments_year, mortgage)
      df["Principal Paid"] = -1 * np.ppmt(interest/payments_year, df.index, years * payments_year, mortgage)
      df["Interest Paid"] = -1 * np.ipmt(interest/payments_year, df.index, years * payments_year, mortgage)
      df = df.round(2)
```

The current DataFrame should look like:

[79]:

Period	Payment Date	Payment	Principal Paid	Interest Paid	Ending Balance
1	2021-01-01	1909.66	576.33	1333.33	NaN
2	2021-02-01	1909.66	578.25	1331.41	NaN
3	2021-03-01	1909.66	580.18	1329.48	NaN
4	2021-04-01	1909.66	582.11	1327.55	NaN
5	2021-05-01	1909.66	584.05	1325.61	NaN
...	...	...	...	...	...
356	2050-08-01	1909.66	1878.15	31.51	NaN
357	2050-09-01	1909.66	1884.41	25.25	NaN
358	2050-10-01	1909.66	1890.69	18.97	NaN
359	2050-11-01	1909.66	1896.99	12.67	NaN
360	2050-12-01	1909.66	1903.32	6.34	NaN

360 rows × 5 columns

Using the Numpy formulas from above, we use **df.index** to quickly populate the DataFrame. As mentioned, it takes some manual subtraction to figure out the Ending Balance, and the first period's *Ending Balance* can be calculated using code below:

[80]:
```
df["Ending Balance"] = 0
df.loc[1, "Ending Balance"] = mortgage - df.loc[1, "Principal Paid"]
```

[81]:
```
df
```

[81]:

Period	Payment Date	Payment	Principal Paid	Interest Paid	Ending Balance
1	2021-01-01	1909.66	576.33	1333.33	399423.67

The **df.loc** property allows us to access a group of rows and columns. We know that the *Ending Balance* is our initial mortgage balance subtract the period's *Principal Paid.* You can probably see already that a simple for loop will be able to do this repetition for us. We just have to keep in mind that the program needs a place to end, and this happens when the *Ending Balance* is 0.

[82]:
```
for period in range(2, len(df)+1):
    previous_balance = df.loc[period-1, 'Ending Balance']
    principal_paid = df.loc[period, "Principal Paid"]

    if previous_balance == 0:
        df.loc[period, ['Payment', 'Principal Paid', 'Interest Paid', 'Ending Balance']] == 0
        continue
    elif principal_paid <= previous_balance:
        df.loc[period, 'Ending Balance'] = previous_balance - principal_paid
```

Implementation of For Loop for Ending Balance Calculation

We didn't include a *Beginning Balance* column, but we include it in the code to calculate *Ending Balance*.

[83]:

Period	Payment Date	Payment	Principal Paid	Interest Paid	Ending Balance
1	2021-01-01	1909.66	576.33	1333.33	399423.67
2	2021-02-01	1909.66	578.25	1331.41	398845.42
3	2021-03-01	1909.66	580.18	1329.48	398265.24
4	2021-04-01	1909.66	582.11	1327.55	397683.13
5	2021-05-01	1909.66	584.05	1325.61	397099.08
...	...	...	...	...	...
356	2050-08-01	1909.66	1878.15	31.51	7575.42
357	2050-09-01	1909.66	1884.41	25.25	5691.01
358	2050-10-01	1909.66	1890.69	18.97	3800.32
359	2050-11-01	1909.66	1896.99	12.67	1903.33
360	2050-12-01	1909.66	1903.32	6.34	0.01

360 rows × 5 columns

At last, our Python calculator is done and our finished product should look like above. Sense check with the Excel one, and we're now ready to tackle other real estate analysis. I like Python for the fact that we don't need to manually count down to 360 rows in Excel, and it will do that for us. Play around with different mortgages, and watch your work come to life.

Saving time and energy is a must, and with this Python code, we can combine it with

other similar projects, whereas Excel is more isolated and standalone. **Congratulations on making it to the end.**

For other Python and Personal Finance projects:

The 4% Rule of Retirement Is Now Obsolete

Here's How to Prepare for Your Retirement and Why You Need Additional Sources of Income.

medium.com

Fundamental Analysis of Stocks for Programmers and Beginners

The crucial accounting and finance topics you need to know to invest like a seasoned professional

medium.com

Your Mortgage Should Not Exceed 2.5x Gross Income

A suitable mortgage amount to take on, and a look at renting vs.

Sign up for Top 10 Stories

By The Startup

Get smarter at building your thing. Subscribe to receive The Startup's top 10 most read stories — delivered straight into your inbox, once a week. Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

Programming Python Mortgage Business Data Science

About Help Legal

Get the Medium app